Appendix C

Transporting VAX C Programs Between VMS and ULTRIX Systems

This appendix describes how to transfer VAX C programs and data between VMS and ULTRIX systems. It also describes many of the differences and incompatibilities that you may need to resolve before a file—developed on one type of system (ULTRIX or VMS) and transported to a different type of system (ULTRIX or VMS)—can be used on the receiving system.

This appendix addresses the following topics:

- Section C.1 describes several methods that you can use to transport VAX C program files between VMS and ULTRIX systems.
- Section C.2 describes differences that result from the compiling and linking process on the two types of systems.

C.1 Transporting VAX C Programs and Other ASCII Files

The following sections describe several mechanisms that you can use to transfer program files between VMS and ULTRIX systems.

C.1.1 Using DECnet-ULTRIX to Copy ASCII Programs

To copy files from a VMS system to an ULTRIX system, you can use the DECnet-ULTRIX[1] copy command, **dcp**, to pull the files over the network. To do this, enter a **dcp** command with the following form:

```
% dcp vms_node/account/password::'dev:[dir]in_file' out_file
```

The definitions of the variables (in italics) in the preceding command line are as follows:

vms_node	VMS node name
account	VMS account that you will log in to
password	Password of the specified VMS account
dev:[dir]	Device (dev) and directory (dir) where the file resides
in_file	Input file name
out_file	Output file name

[1] DECnet-ULTRIX is a layered product available under a separate license for ULTRIX systems.

To copy files from an ULTRIX node to a VMS node, use the **dcp** command to push the files over the network. To do this, enter a **dcp** command with the following form:

```
% dcp in_file vms_node/account/password::'dev:[dir]out_file'
```

The variables in this command line are the same as those in the previous command line.

C.1.2 Using DECnet–VAX to Copy ASCII Programs

You can use the DECnet–VAX[2] COPY command to copy ASCII files between a VMS node and an ULTRIX node. To pull a VAX C source file from an ULTRIX node to a VMS node, enter the COPY command with the following form:

```
$ COPY ult_node"account password"::"/dev/dir/in_file" out_fil e
```

The definitions of the variables (in italics) in the preceding command line are as follows:

ult_node	ULTRIX node name
account	ULTRIX account that you will log in to
password	Password of the specified ULTRIX account
/dev/dir/	Device (dev) and directory (dir) where the file resides
in_file	Input file name
out_file	Output file name

The resulting VMS file is not created with record format VARIABLE. This causes the VMS editor (EDT) to issue the following message:

```
Input file does not have standard text file format
```

Otherwise, the file can be used as an ASCII file.

C.1.3 DEC/Shell on a VMS System — The tar Utility

You can use the DEC/Shell[3] tar utility to move ASCII files from a VMS system to an ULTRIX system, or to restore ASCII files written by tar on an ULTRIX system. When restoring files from an ULTRIX system to a VMS system, you must physically mount the tape on the tape drive from which you want to perform the restore procedure. You do not need to enter the MOUNT command because the tar utility does this automatically.

NOTE

> The tar utility cannot be used alone to transfer binary data files between a VMS system and an ULTRIX system. The data formats on the two systems are incompatible and require additional processing beyond that provided by tar.

In the following example, the function letter **c** directs the tar utility to create a new tape. Writing starts at the beginning of the tape instead of after the last file. The tar utility writes the named file(s) to the tape.

```
% tar -c foo.c
```

[2] DECnet–VAX is a layered product available under a separate license for VMS and MicroVMS systems.

[3] DEC/Shell is a layered product available for VMS and MicroVMS systems under a separate license.

The following command extracts all the files in the ./cprogs directory. By default, the device mta0: is used.

```
% tar -x ./cprogs
```

The function letter **x** directs tar to extract files from tape. Because a directory name is given as a parameter, tar recursively extracts files from the directory. If you do not specify a parameter, tar extracts the entire contents of the tape.

C.2 Compiling and Linking Considerations

The following sections describe the I/O files associated with the **vcc** (on ULTRIX systems) and **CC** (on VMS systems) compilation commands, the search paths associated with the **vcc** command, and the differences in psect usage and image sizes on the two types of systems.

C.2.1 Input and Output Files

On a VMS system, the process of compiling and linking a VAX C program generally requires two steps. For example, if you have a VAX C source program named FOO.C, the following sequence of commands results in an executable image, FOO.EXE:

```
$ CC FOO
$ LINK FOO
```

The first command invokes the VAX C compiler, which compiles the input file FOO.C, and produces the output file FOO.OBJ. The second command invokes the linker, which links the input file FOO.OBJ (along with appropriate support routines), and produces the output file FOO.EXE.

On an ULTRIX system, you can accomplish the same process by using a single **vcc** command line. The **vcc** command causes the VAX C compiler and the linker to execute with the appropriate arguments, based on its interpretation of its own arguments. Other processors (such as the ULTRIX assembler and the ULTRIX C preprocessor, cpp) can also be invoked by the **vcc** command, depending on the arguments supplied on the **vcc**command line.

The following **vcc** command is similar to the VMS command sequence previously described:

```
% vcc -o foo foo.c
```

In this example, the presence of foo.c causes the **vcc** command to invoke the VAX C compiler with the file foo.c as an input file; the compiler produces the file foo.o as output. The .c must be supplied explicitly. The **vcc** command then invokes the linker with –o foo and foo.o (and appropriate libraries) as arguments, and the linker produces the executable image foo. The **vcc** command program then deletes the intermediate file foo.o.

The **vcc** command assumes the existence of intermediate files not specified by the command line, which are created by one processor and passed to another (for example, foo.o, created by the VAX C compiler and linked by the linker). The **vcc** command derives the names of these intermediate files from its arguments. For example, the **vcc** command assumes that any file whose name ends in .c or .h is a VAX C source file that should be compiled by the VAX C compiler. The **vcc** command further assumes that the VAX C compiler will create a file of the same name (minus the .c or .h), with the file extension .o. For example, compiling foo.c produces foo.o.

C.2.2 Search Paths Used by the vcc Command

The **vcc** command looks for the processor it expects to execute, as well as support routine libraries and objects, in a sequence of directories in the ULTRIX file system. This sequence, or search path, is as follows:

- The **vcc** command looks for the VAX C compiler by trying to execute in order:
  1. The directory specified in the **–B** option, if **–t0** is specified.
  2. /usr/lib/vcc
  3. /lib/vcc
- The **vcc** command looks for the linker in the following places:
  1. The directory specified in the **–B** option, if **–tl** is specified.
  2. /usr/bin
  3. /bin
- If the **–Em** option is specified, the **vcc** command looks for the ULTRIX C preprocessor cpp in the following directories:
  1. The directory specified in the **–B** option, if **–tp** is specified.
  2. /usr/lib
  3. /lib
- The **vcc** command assumes that the C libraries are in the /usr/lib directory.

If you need to print a diagnostic message while executing either the VAX C compiler, the linker, or a user program, the message routines look for the following messages in the file:

- $fortmsgfile — if the environment variable fortmsgfile is defined
- /usr/lib/fortmsgfile — otherwise

C.2.3 Psect Differences

There are numerous differences between the way that psects are set up by VAX C on a VMS system and the way that they are set up by VAX C on an ULTRIX system.

C.2.4 Image Size Differences

The size of the executable VAX C programs is much larger on ULTRIX systems than with either VAX C programs on VMS systems or pcc programs on ULTRIX systems. There are several reasons for this as follows:

- The VMS operating system supports shareable images, in which libraries of subroutines can be shared by more than one program. This effectively reduces the size of a program's executable image for those programs that use routines contained within the shared libraries. For VAX C programs, these shareable libraries include the math library routines, Record Management Services (RMS) routines (on VMS systems), and system service routines.

Unlike the VMS operating system, the ULTRIX operating system has no comparable facility for sharing common routines; each program that uses shared libraries or system service routines must have the routines physically present as part of the program's executable image. As a result, these images are much larger on an ULTRIX system than their counterparts would be on a VMS system.

- The ULTRIX system performs demand zero compression only on the bss section of a program. (The bss section appears at the end of a program image file.) When image activation occurs on an ULTRIX system, this section is allocated zero initialized memory.

  Demand-zero compression is the extraction of contiguous, uninitialized, writeable pages from an image section and the placing of these pages into a newly created demand-zero image section.

  A demand-zero image section contains uninitialized, writeable pages that do not occupy space in the image file on disk, but which, when accessed during program execution, are allocated memory and initialized with binary zeros by the operating system.

  In pcc, the compiler does the work of separating initialized data from uninitialized data. However, many of the VAX C optimizations depend on a storage layout that prevents such separation. For this reason, no local storage for variables or arrays is allocated to bss.

C.3 Transferring Data Files Between VMS and ULTRIX Systems

You can make data file transfers between VMS and ULTRIX systems by using the DECnet dcp utility described in Section C.1.1 or magnetic tape, using the tape archive utility, tar, which is described in Section C.1.3.